Bewertetes Übungsblatt 1

Sommersemester 2018

Veranstaltung: Wirtschaftsinformatik 2

Lukas Reuter (reuter@uni-trier.de), Prof. Dr. Ingo J. Timm
Lehrstuhl für Wirtschaftsinformatik I, Universität Trier

1 Informierte Suchalgorithmen

Im Rahmen der Vorlesung haben Sie bereits das *8-Puzzle* (siehe Abbildung 1) kennen gelernt. Auf einem 3x3 Felder großen Spielfeld sind Kacheln mit den Zahlen von 1 bis 8 verteilt. Ziel des Spiels ist es, durch das verschieben der Kacheln auf das jeweils leere Feld den gezeigten Zustand, in dem die Zahlen geordnet sind, zu erreichen. Soll dieses Problem mit der Unterstützung von Computern gelöst werden, ergibt sich ein rechenintensives Suchproblem.

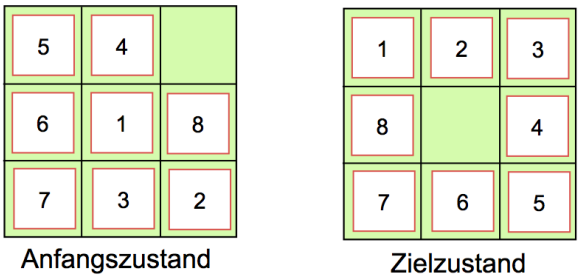

Abbildung 1: Zustände eines 8-Puzzles

Implementieren Sie das *8-Puzzle* in Java. Nutzen Sie eine ressourceneffiziente Datenstruktur zur Verwaltung der möglichen Zustände des Puzzles (gegeben ist ein zweidimensionales Array) und wenden Sie die folgenden Suchalgorithmen an, um eine Lösung zu generieren. Als Grundlage für Ihre Implementierung nutzen Sie bitte die in StudIP zur Verfügung gestellten Java-Klassen (*puzzleSolver.zip*). Die Hauptklasse (*puzzleSolver.java*) soll über die Konsole ausführbar sein, also müssen die relevanten Parameter über das String-Array args der *main()*-Methode übergeben werden. Hierbei sind die Parameter wie folgt zu verwenden:

- args[0]: Hier soll das Puzzle, mit Komma als Trennzeichen oder direkt als *int[][]*-Array, übergeben werden können. (z.B.: 1,2,3,4,5,6,7,8,0) Die Zahl 0 dient als Platzhalter, um das leere Feld zu repräsentieren.
- args[1]: Dieser Parameter soll den Suchalgorithmus variieren. (1: greedy, 2: A* mit h(n) = falsch platzierten Kacheln, 3: A* mit h(n) = Manhattan-Distanzen)
- args[2]: Über einen Debugging-Parameter soll gewählt werden können, ob eine vollständige Ausgabe der Suchschritte stattfindet. (0: nein, 1: ja)

Innerhalb der *main()*-Methode legen Sie dann das Puzzle als Objekt des Typs *puzzle* an - die entsprechende Klasse finden Sie in der Datei *puzzle.java*. Für die Nachvollziehbarkeit Ihres Algorithmus ist es wichtig, dass der Lösungsweg anschaulich ausgegeben wird. Hierzu dient die Methode *printPuzzle()*, mit der Sie nach jeder Iteration des Algorithmus den aktuellen Zustand des Puzzles ausgeben können. Aus Performancegründen ist jedoch die Ausgabe von Zweigen des Suchbaum, welche nicht zielführend sind, nicht sinnvoll. Deshalb soll dies nur dann geschehen, wenn der o.g. Debugging-Parameter entsprechend gesetzt wurde. Anderenfalls ist lediglich die Ausgabe des korrekten/besten Lösungsweges notwendig. Beachten Sie bitte, dass die Erweiterung der Ausgabe der *printPuzzle()*-Methode um die aktuellen Werte der Funktionen h(n), g(n) und f(n) sinnvoll ist.

Sollten Sie auf Grund der Gleichheit von Werten der Kostenfunktionen keine eindeutige Wahl treffen können, wählen Sie den Knoten (die Kachel) mit der niedrigsten Ziffer aus.

WI2— Bewertetes Übungsblatt 1

1.1 Greedy-Search

Versuchen Sie zunächst mit Hilfe von Greedy-Search das *8-Puzzle* zu lösen. Verwenden Sie als Kostenfunktion $h(n)$ die Anzahl der falsch platzierten Kacheln. Dokumentieren Sie Ihre Ergebnisse, was fällt ihnen auf?

1.2 A*-Algorithmus

Implementieren Sie nun den A*-Algorithmus zur Lösungsfindung. Die Gesamtkosten f(n) sollen sich hierbei aus der Summe der bisherigen Kosten g(n), welche sich aus der Anzahl der bereits durchgeführten Spielzüge ergibt, und den geschätzten Kosten h(n) bis zum Zielzustand berechnen. Unterscheiden Sie hierbei in folgende Heuristiken zur Ermittlung von h(n):

1. $h_1(n)$ = Anzahl der Kacheln in falscher Position
2. $h_2(n)$ = Manhattan-Distanz[1] der Kacheln zum Endzustand

Vergleichen und bewerten Sie Ihre Ergebnisse. Identifizieren Sie zudem Optimierungspotenziale und beschreiben Sie kurz mögliche Modifikationen des Suchalgorithmus, mit denen die Suche optimiert werden könnte.

Die Abgabe dieser Übung kann in Gruppen von bis zu 5 Personen erfolgen. Geben Sie in der Abgabe alle Gruppenmitglieder mit Namen und Matrikelnummer an. Für die Klausurzualssung ist die Abgabe sowie die erfolgreiche Bearbeitung dieser Übung zwingend erforderlich. Bitte nutzen Sie *Java* als Programmiersprache und laden Sie den Quellcode gepackt als ZIP- oder RAR-Archiv in den entsprechenden Ordner in StudIP hoch.
Abgabefrist ist **Sonntag, der 13. Mai 2018 um 23:59 Uhr**.

Anmerkungen zur Abgabe: Ihr Programm wird mit unterschiedlichen Beispielen getestet. Achten Sie darauf, dass ihr Programm mit beliebigen Eingaben ein korrektes Ergebnis liefert. Um das Verständnis für den Quellcode zu erleichtern, kommentieren Sie diesen bitte aussagekräftig. Fertigen Sie zudem eine Dokumentation Ihres Programms in Form eines schriftlichen Begleitschreibens (PDF) an.

[1] Die Manhattan-Distanz zwischen zwei Punkten berechnet sich aus Summe der absoluten Differenzen der Einzelkoordinaten. Im vorliegenden Beispiel also aus der minimalen Anzahl der Züge die benötigt werden, bis eine Kachel an der korrekten Endposition liegt. $d(a,b) = |a1 - b1| + |a2 - b2|$